cuPyNumeric and Legate

Manolis Papadakis, Software Engineering Manager, Nvidia

Scaling NumPy code with cuPyNumeric

Simple Python code w/ NumPy
No partitioning code
No MPI code

```
import cupynumeric as np

m, n, k = 1000000, 1000000, 1000000 # 1Mx1M
A = np.random.rand(m,k).astype(np.float32)
B = np.random.rand(k,n).astype(np.float32)

res = A @ B
```

Get your science done faster!

- **Prototype quickly** in the NumPy API.
- **Scale** from a single CPU core to a multi-GPU multi-node supercomputer
- **Use HPC clusters without HPC programming** or expensive rewrites in e.g. MPI.
- **Implements most NumPy functionality**, with growing support for SciPy, and other scientific libraries.

Scales to multi-GPU multi-Node at runtime

```
$ legate --launcher mpirun --nodes 64 --gpus 8 matmul.py
```

Legate Ecosystem

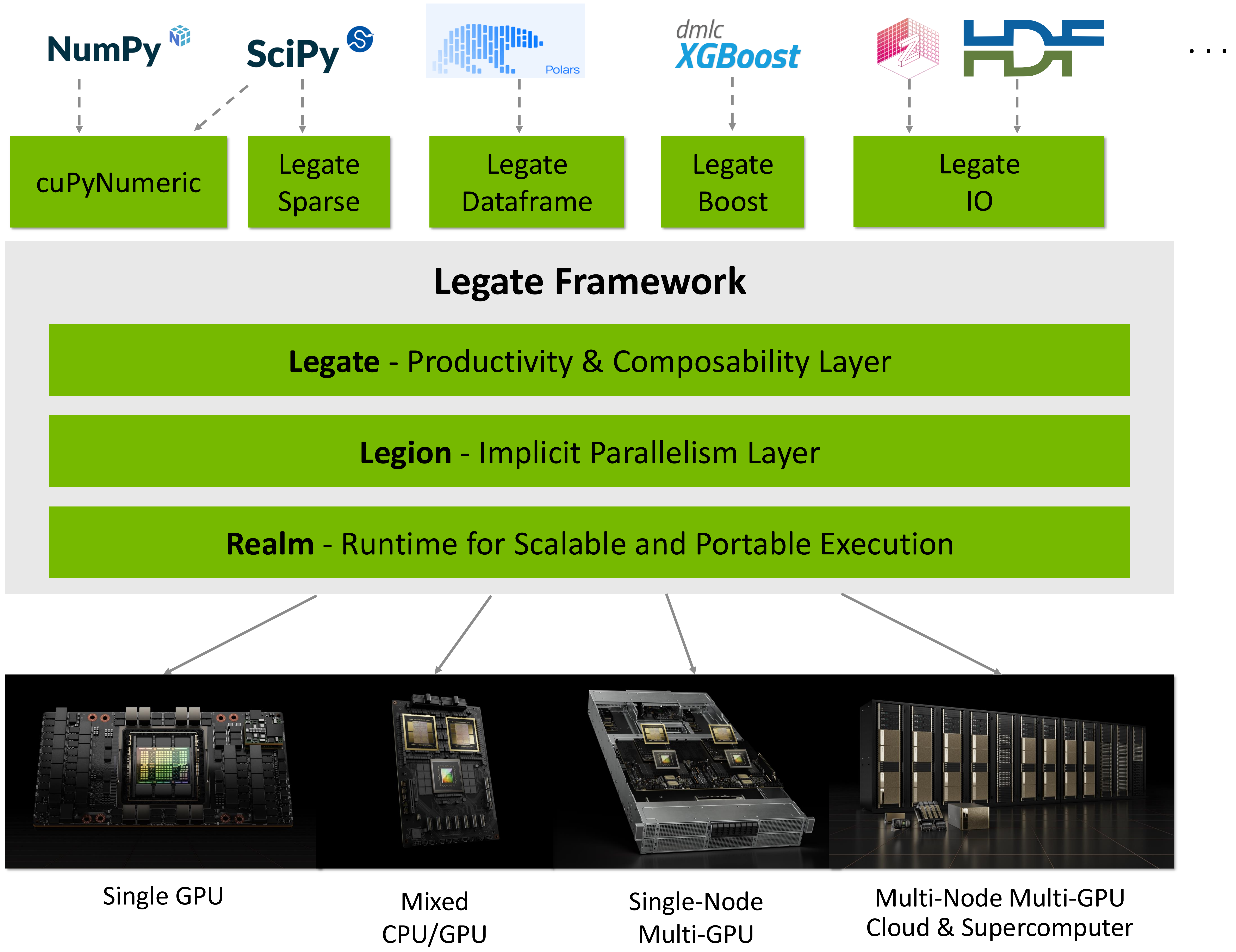
Transparently scalable libraries, built on a common foundation

Productive libraries for transparent scaling

- Author programs in familiar APIs, scale to any machine
- Easy transition from prototyping to production
- Common runtime enables composability

Powered by a common distributed execution framework

- Task-based, with implicit parallelism
- Provides common distributed data format, that allows zero-copy data passing between libraries
- Decides degree of parallelism, data partitioning and task placement (following library-specified partitioning constraints)
- Manages synchronization, data movement and coherence (following library-specified usage annotations - read, write, reduce)


NumPy
SciPy
Polars
dmlc XGBoost
HDF
. . .
cuPyNumeric
Legate Sparse
Legate Dataframe
Legate Boost
Legate IO
Legate Framework
Legate - Productivity & Composability Layer
Legion - Implicit Parallelism Layer
Realm - Runtime for Scalable and Portable Execution
Single GPU
Mixed CPU/GPU
Single-Node Multi-GPU
Multi-Node Multi-GPU Cloud & Supercomputer